learning.py为python代码运行助手。

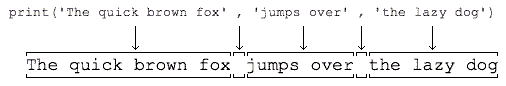
**输出**

print()函数也可以接受多个字符串，用逗号“,”隔开，就可以连成一串输出：

>>> print('The quick brown fox', 'jumps over', 'the lazy dog')

The quick brown fox jumps over the lazy dog

print()会依次打印每个字符串，遇到逗号“,”会输出一个空格，因此，输出的字符串是这样拼起来的：



### 输入

程序运行的时候，没有任何提示信息告诉用户：“嘿，赶紧输入你的名字”，这样显得很不友好。幸好，input()可以让你显示一个字符串来提示用户，于是我们把代码改成：

name = input('please enter your name: ')

print('hello,', name)

#### 字符串

转义字符\可以转义很多字符，比如\n表示换行，\t表示制表符，字符\本身也要转义，所以\\表示的字符就是\，可以在Python的交互式命令行用print()打印字符串看看：

>>> print('I\'m ok.')

I'm ok.

>>> print('I\'m learning\nPython.')

I'm learning

Python.

>>> print('\\\n\\')

\

\

如果字符串里面有很多字符都需要转义，就需要加很多\，为了简化，Python还允许用r''表示''内部的字符串默认不转义，可以自己试试：

>>> print('\\\t\\')

\ \

>>> print(r'\\\t\\')

\\\t\\

如果字符串内部有很多换行，用\n写在一行里不好阅读，为了简化，Python允许用'''...'''的格式表示多行内容，可以自己试试：

>>> print('''line1

... line2

... line3''')

line1

line2

line3

上面是在交互式命令行内输入，注意在输入多行内容时，提示符由>>>变为...，提示你可以接着上一行输入，注意...是提示符，不是代码的一部分：

┌────────────────────────────────────────────────────────┐

│Command Prompt - python \_ □ x │

├────────────────────────────────────────────────────────┤

│>>> print('''line1 │

│... line2 │

│... line3''') │

│line1 │

│line2 │

│line3 │

│ │

│>>> \_ │

│ │

│ │

│ │

└────────────────────────────────────────────────────────┘

当输入完结束符```和括号)后，执行该语句并打印结果。

如果写成程序并存为.py文件，就是：

print('''line1

line2

line3''')

多行字符串'''...'''还可以在前面加上r使用，请自行测试：

print(r'''hello,\n

world''')

#### 布尔值

在Python中，可以直接用True、False表示布尔值（请注意大小写），也可以通过布尔运算计算出来：

>>> True

True

>>> False

False

>>> 3 > 2

True

>>> 3 > 5

False

布尔值可以用and、or和not运算。

#### 空值

空值是Python里一个特殊的值，用None表示。None不能理解为0，因为0是有意义的，而None是一个特殊的空值。

### 变量

变量在程序中就是用一个变量名表示了，变量名必须是大小写英文、数字和\_的组合，且不能用数字开头。

### 常量

在Python中，通常用全部大写的变量名表示常量：

PI = 3.14159265359

但事实上PI仍然是一个变量，Python根本没有任何机制保证PI不会被改变，所以，用全部大写的变量名表示常量只是一个习惯上的用法，如果你一定要改变变量PI的值，也没人能拦住你。

### 除法

在Python中，有两种除法，一种除法是/：

>>> 10 / 3

3.3333333333333335

/除法计算结果是浮点数，即使是两个整数恰好整除，结果也是浮点数：

>>> 9 / 3

3.0

还有一种除法是//，称为地板除，两个整数的除法仍然是整数：

>>> 10 // 3

3

你没有看错，整数的地板除//永远是整数，即使除不尽。要做精确的除法，使用/就可以。

因为//除法只取结果的整数部分，所以Python还提供一个余数运算，可以得到两个整数相除的余数：

>>> 10 % 3

1

无论整数做//除法还是取余数，结果永远是整数，所以，整数运算结果永远是精确的。

注意：Python的整数没有大小限制，而某些语言的整数根据其存储长度是有大小限制的，例如Java对32位整数的范围限制在-2147483648-2147483647。

Python的浮点数也没有大小限制，但是超出一定范围就直接表示为inf（无限大）。

### 格式化

在Python中，采用的格式化方式和C语言是一致的，用%实现，举例如下：

>>> 'Hello, %s' % 'world'

'Hello, world'

>>> 'Hi, %s, you have $%d.' % ('Michael', 1000000)

'Hi, Michael, you have $1000000.'

你可能猜到了，%运算符就是用来格式化字符串的。在字符串内部，%s表示用字符串替换，%d表示用整数替换，有几个%?占位符，后面就跟几个变量或者值，顺序要对应好。如果只有一个%?，括号可以省略。

常见的占位符有：

| **占位符** | **替换内容** |
| --- | --- |
| %d | 整数 |
| %f | 浮点数 |
| %s | 字符串 |
| %x | 十六进制整数 |

如果你不太确定应该用什么，%s永远起作用，它会把任何数据类型转换为字符串。

有些时候，字符串里面的%是一个普通字符怎么办？这个时候就需要转义，用%%来表示一个%。

### format()

另一种格式化字符串的方法是使用字符串的format()方法，它会用传入的参数依次替换字符串内的占位符{0}、{1}……，不过这种方式写起来比%要麻烦得多：

>>> 'Hello, {0}, 成绩提升了 {1:.1f}%'.format('小明', 17.125)

'Hello, 小明, 成绩提升了 17.1%'

### list

Python内置的一种数据类型是列表：list。list是一种有序的集合，可以随时添加和删除其中的元素。

比如，列出班里所有同学的名字，就可以用一个list表示：

>>> classmates = ['Michael', 'Bob', 'Tracy']

>>> classmates

['Michael', 'Bob', 'Tracy']

变量classmates就是一个list。用len()函数可以获得list元素的个数：

>>> len(classmates)

3

用索引来访问list中每一个位置的元素，记得索引是从0开始的：

>>> classmates[0]

'Michael'

>>> classmates[1]

'Bob'

>>> classmates[2]

'Tracy'

>>> classmates[3]

如果要取最后一个元素，除了计算索引位置外，还可以用-1做索引，直接获取最后一个元素：

>>> classmates[-1]

'Tracy'

以此类推，可以获取倒数第2个、倒数第3个：

>>> classmates[-2]

'Bob'

>>> classmates[-3]

'Michael'

list是一个可变的有序表，所以，可以往list中追加元素到末尾：

>>> classmates.append('Adam')

>>> classmates

['Michael', 'Bob', 'Tracy', 'Adam']

也可以把元素插入到指定的位置，比如索引号为1的位置：

>>> classmates.insert(1, 'Jack')

>>> classmates

['Michael', 'Jack', 'Bob', 'Tracy', 'Adam']

要删除list末尾的元素，用pop()方法：

>>> classmates.pop()

'Adam'

>>> classmates

['Michael', 'Jack', 'Bob', 'Tracy']

要删除指定位置的元素，用pop(i)方法，其中i是索引位置：

>>> classmates.pop(1)

'Jack'

>>> classmates

['Michael', 'Bob', 'Tracy']

要把某个元素替换成别的元素，可以直接赋值给对应的索引位置：

>>> classmates[1] = 'Sarah'

>>> classmates

['Michael', 'Sarah', 'Tracy']

list里面的元素的数据类型也可以不同，比如：

>>> L = ['Apple', 123, True]

list元素也可以是另一个list，比如：

>>> s = ['python', 'java', ['asp', 'php'], 'scheme']

>>> len(s)

4

要注意s只有4个元素，其中s[2]又是一个list，如果拆开写就更容易理解了：

>>> p = ['asp', 'php']

>>> s = ['python', 'java', p, 'scheme']

要拿到'php'可以写p[1]或者s[2][1]，因此s可以看成是一个二维数组，类似的还有三维、四维……数组，不过很少用到。

如果一个list中一个元素也没有，就是一个空的list，它的长度为0：

>>> L = []

>>> len(L)

0

### tuple

另一种有序列表叫元组：tuple。tuple和list非常类似，但是tuple一旦初始化就不能修改。

不可变的tuple有什么意义？因为tuple不可变，所以代码更安全。如果可能，能用tuple代替list就尽量用tuple。

tuple的陷阱：当你定义一个tuple时，在定义的时候，tuple的元素就必须被确定下来，比如：

>>> t = (1, 2)

>>> t

(1, 2)

如果要定义一个空的tuple，可以写成()：

>>> t = ()

>>> t

()

但是，要定义一个只有1个元素的tuple，如果你这么定义：

>>> t = (1)

>>> t

1

定义的不是tuple，是1这个数！这是因为括号()既可以表示tuple，又可以表示数学公式中的小括号，这就产生了歧义，因此，Python规定，这种情况下，按小括号进行计算，计算结果自然是1。

所以，只有1个元素的tuple定义时必须加一个逗号,，来消除歧义：

>>> t = (1,)

>>> t

(1,)

Python在显示只有1个元素的tuple时，也会加一个逗号,，以免你误解成数学计算意义上的括号。

最后来看一个“可变的”tuple：

>>> t = ('a', 'b', ['A', 'B'])

>>> t[2][0] = 'X'

>>> t[2][1] = 'Y'

>>> t

('a', 'b', ['X', 'Y'])

表面上看，tuple的元素确实变了，但其实变的不是tuple的元素，而是list的元素。tuple一开始指向的list并没有改成别的list，所以，tuple所谓的“不变”是说，tuple的每个元素，指向永远不变。即指向'a'，就不能改成指向'b'，指向一个list，就不能改成指向其他对象，但指向的这个list本身是可变的！

理解了“指向不变”后，要创建一个内容也不变的tuple怎么做？那就必须保证tuple的每一个元素本身也不能变。

条件判断条件判断可以让计算机自己做选择，Python的if...elif...else很灵活。

条件判断从上向下匹配，当满足条件时执行对应的块内语句，后续的elif和else都不再执行。

